Andrew Glassner's Notebook

http://www.glassner.com

Digital Weaving, Part 3

Andrew Glassner

In my last two columns I described the theory and practice of weaving, and I presented a language for creating woven patterns. This time I'll talk about the important and interesting fabrics known as tartans, and then discuss writing a program to simulate a loom.

A quick review

Recall from part 1 that a *twill* is a distinctive pattern type, formed by a tie-up that creates a diagonal design. Perhaps the best-known twill fabric today is denim, but close behind that is the Scottish *tartan*. Tartans are the familiar-colored plaids that have traditionally adorned kilts and other formal Scottish clothing.

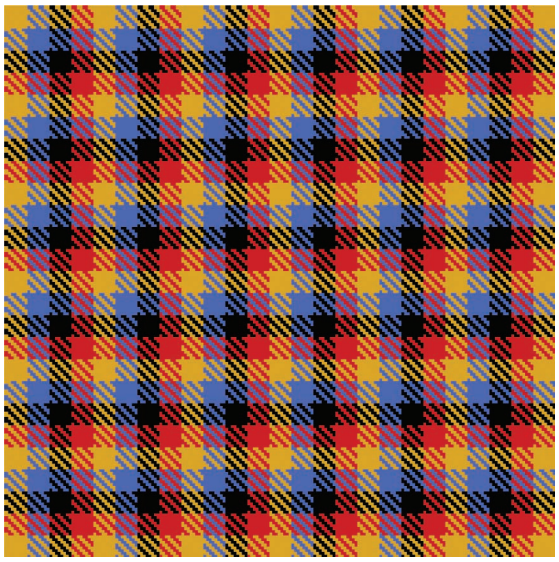
Tartans always use the same pattern of colors in the warp and the weft; weavers say that such a pattern is "tromp as writ" (for complete discussions of weaving terms and ideas, please see the previous installments of my column in the November/December 2002 and January/February 2003 issues). The result is that tartans are made up of intersecting stripes of different widths and colors. As Figure 1 shows, when two stripes of different colors overlap, we get a blended color that mixes equal amounts of each. When two stripes of the same color overlap, we get a solid rectangle of that color. The color pattern that describes a particular tartan is called its *sett*.

Today's tartan

Today weavers regularly produce thousands of different setts. Tartan cloth is used casually and for ceremonial clothing, kilts, and other formal wear. Because some clans and families are protective of their registered tartans, there arose a need for some shared patterns that people could wear when they gathered as a group. For example, if a traditional marching band got together and everyone wore their own family's sett, it could look like a crazy mishmash of patterns. To help unify groups that don't have their own tartan, three setts (Hunting Stewart, Caledonia, and Black Watch) have been generally accepted as universal tartans that may be worn by one and all.

The Hunting Stewart pattern (see Figure 2) has been around since the early 1800s. The Caledonia sett (Figure 3) probably dates back to about 1800, and members of the British Army have worn the Black Watch (or Government) sett (Figure 4) since the 1740s.

Tartan colors are generally drawn from a palette of a couple of dozen colors, with custom colors occasionally included for specific setts. Most colors are referred to with labels that consist of just a few letters. I'm not aware of any standardized RGB list of these colors, so I examined and measured several hundred tartans and compared their actual colors against their labels. Some variation exists, of course, but the colors are pretty consistent across different manufacturers and sources. Table 1 summarizes

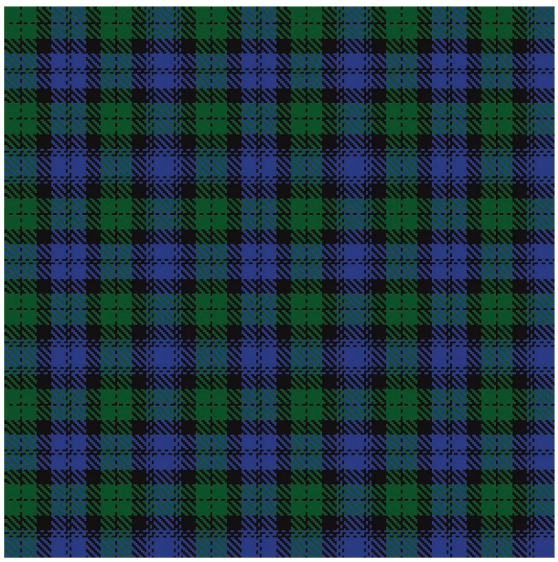


1 In a twill, when two stripes of different colors overlap, we see an equal mix of their colors. When the two stripes have the same color, we see a solid block of that color.

2 The Hunting Stewart tartan. The AWL (recall that AWL is Andrew's Weaving Language, which I described in detail in part 2) expression for this sett is B 9 G 4 B 9 K 3 B 3 K 8 G 27 R 4 G 27 K 8 G 5 K 13 G 4 K 13 G 5 K 8 G 27 Y 4 G 27 K 8 B 3 K 3 tartan.

Published by the IEEE Computer Society

3 The Caledonia tartan. The AWL expression for this sett is R 42 A 18 K 4 A 4 K 4 A 18 K 36 Y 6 G 42 R 26 K 6 R 26 W 4 R 26 tartan.



4 The Black Watch tartan. The AWL expression for this sett is B 22 K 2 B 2 K 2 B 2 K 16 G 16 K 2 G 16 K 16 B 16 K 2 B 2 tartan.

Table 1. RGB values for tartan colors, identified by their traditional initials and a more informative label.

Initial	R	G	B	Name
A	60	132	172	Aqua
B	44	58	132	Blue
DB	12	10	76	Dark blue
LB	124	130	196	Light blue
MB	20	26	68	Dark blue
NB	4	2	36	Navy blue
RB	4	2	100	Royal blue
C	148	2	36	Dark pink
VLC	220	170	172	Pink
DG	4	50	20	Dark green
FG	68	106	84	Blue-green
G	4	82	36	Green
LG	44	154	20	Light green
MG	4	58	20	Darker green
K	20	18	20	Black
M	116	26	52	Magenta
DN	76	74	76	Dark gray
LN	188	186	188	Light gray
N	124	122	124	Mid-gray
DO	220	90	4	Dark orange
LO	236	114	60	Light orange
O	252	74	4	Orange
P	116	2	116	Purple
DR	204	2	4	Dark red
LR	204	42	44	Light red
R	204	2	4	Red
WR	100	2	44	Dark magenta
S	228	86	4	Sandy yellow
DT	68	18	4	Dark tan
LPT	204	150	100	Slightly light tan
LT	148	102	52	Light tan
RT	244	90	44	Reddish tan
T	84	62	20	Tan
MU	204	122	20	Orangeish
W	228	226	228	White
DY	148	122	4	Dark yellow
LY	244	218	4	Light yellow
Y	236	194	4	Yellow

the tartan colors. Note that white and black aren't at the limits of the RGB range. This is because natural wools have a more limited color gamut than CRT phosphors.

Building a digital loom

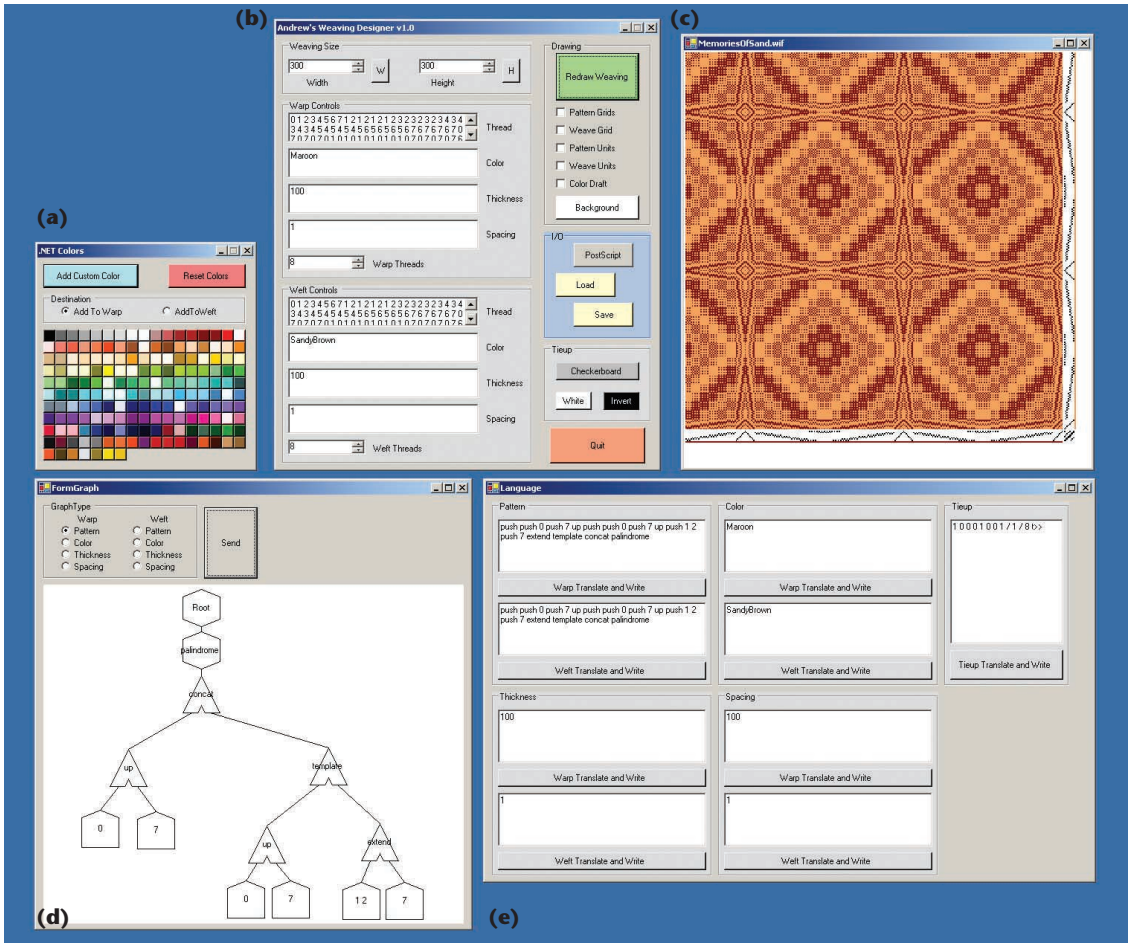
I built my digital loom using C#, a new programming language from Microsoft. For many years I've been happily programming in good, old-fashioned, vanilla C.

I decided to use my digital loom as an immigration project into the C# language and its associated .NET environment for writing Windows code. I was surprised by how easy it was to write my program in this system, and in the course of this one project I've become a C# convert.

Let's look at the digital loom from the outside–in, starting with the user interface. Figure 5 (next page) shows a screen shot of the digital loom in action. This figure includes a graphical editor window that I didn't include in the screen shot from part 2 of this column. I built this because I thought it might be fun to enter AWL expressions by drawing them, rather than typing them in.

The main control form in Figure 5b and the woven fabric output in Figure 5c were the only two windows I had for quite a while. The weaving display shows the warp and weft patterns and the tie-up using traditional black-and-white matrices, with thread colors drawn outside them. This is mostly a read-only display, but you can flip bits in the tie-up by clicking on them.

The panel in Figure 5b holds most of the basic controls. The upper left of this panel has counters to set the width and height of the displayed fabric in the weaving window. The other major controls are buttons to speci-



5 A screen shot of my digital loom. Figure 5a shows the color chips. Figure 5b is the main control panel, and the window in Figure 5c holds the woven fabric. Figure 5d is my graphical AWL editor, and Figure 5e is the AWL evaluation window.

fy some details about the display itself, saving and loading weaving files in the weaving information file (WIF) format, and a button to save the weaving in PostScript. The left side of this panel has two sets of four text boxes. Each set (one for the warp threads, one for the weft) lets me enter expressions for the threading pattern, thread colors, thickness, and spacing. There's also a numerical counter for identifying how many threads are to be used for that pattern.

Because I specify colors by name in the text fields (such as black or aqua), I provided a little color selector panel as well, shown in the upper right. It contains all the default colors in the .NET environment, as well as all the tartan colors named in Table 1. You can also create your own colors and add them to the list. If you click on a color chip, its name is appended to the list of colors in either the warp or weft window, depending on which of the two radio buttons you've selected.

For a long time this was all I had. Then when I wrote my intrepreter for AWL, I created the new form in Figure 5e (recall that AWL is Andrew's Weaving Language, which I described in detail in part 2). This has nine text boxes. The first eight correspond one-to-one to the text boxes in the main panel. The difference is that in the main panel you enter explicit numerical patterns for the warp and weft, whereas in the AWL windows you enter AWL expressions. By clicking the button beneath each box, the system translates the AWL into its resulting pattern, and copies that into the corresponding box in the main form. That way you can see what your expression translates to, while the AWL is still there and editable. A ninth box on this form lets you enter an AWL expression for the tieup.

I thought it might be fun to write a graphical editor for AWL expressions, and you can see that window in Figure 5d. Like most such editors, you can create and delete nodes, drag them around, change the wiring, and so on. Once you have a drawing you like, you identify which AWL expression window you want to send it to, and push the send button to translate the drawing into AWL. If you like the expression you've created, then you press the button below that window as usual to evaluate it and pass it to the main form. The graphical editor is interesting to play with, but I eventually found that I preferred to type and edit my AWL expressions directly. You can see in Figure 5c the result of the expression drawn in the graphical editor. The expression contains some redundant instances of the push command, which doesn't hurt things (nothing happens if there's nothing new to push).

Let's move inside the code now. Programming this digital loom was straightforward. Of course, the final code's structure reflects that I learned (the C# language) as I went, but it works fine and the tasks are simple enough that everything happens effectively, even with a naive and blunt programming style.

Program highlights

I'll discuss the three most interesting bits of the program: drawing the fabric, reading and writing WIF files, and evaluating AWL expressions.

To draw the fabric, I start by finding the current window size (since you can grab it at a corner and make it bigger and smaller), and I compare this to the fabric size that I want to draw inside of it. As Figure 6 shows, suppose the weaving is a cells high, and we need b boxes to hold the pattern along the bottom. I need to also include space equal to one box at the top for a border, one between the weaving and the pattern, and one more at the bottom for a border. I also use a half-cell of space between the weaving pattern and the thread colors, resulting in a height of $a + b + 3.5$ cells. I then use the same layout to find the number of cells that are required horizontally.

From this I can find the largest square's size that I can use for a cell that still fits in the window. Even though the spacing specification for a given weave can change the location and aspect ratio of the cells in the weave itself, I treat everything as squares at this point.

Filling in the cells for the tie-up, pattern grids, and row or column of colors is straightforward. I just draw the grids and fill in the boxes where necessary—either with the thread color or black and white for the patterns and tie-up.

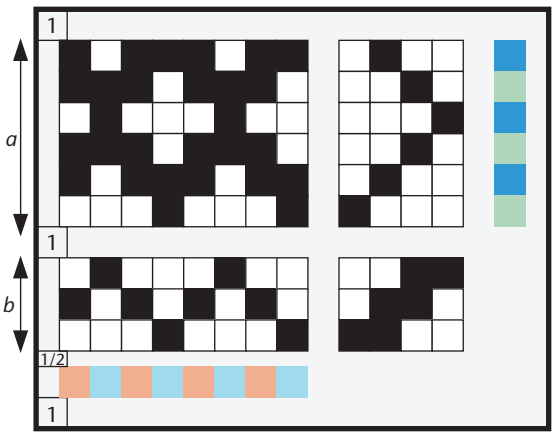
To draw the weaving itself, I proceed one cell at a time. First I use the spacing information to determine the cell location and size. Then I consult the thickness and color for the threads at this cell to find the two rectangles that fit into the cell. Finally, I consult the warp, weft, and tie-up patterns using my weaving equation (presented in part 1) to determine which rectangle is on top. I draw the two rectangles in the proper order, and then move on to the next cell.

The WIF standard defines the usual way to save weaving information. Adhering to WIF means that my system can trade weaving files with the commercially available digital weaving systems, and I can also read the many WIF files available on the Web.

WIF files are easy to write, but can be tricky to read. The WIF standard lets you write out sections, or blocks, of information in any order, so information that you need to parse or interpret for one part of the file might not appear until much later. Some sections don't need to exist at all.

There are two general approaches to handling this sort of thing. The more efficient way is to first create a list of pointers into the file that indicate where every possible section starts. If you save these as integer offsets from the start of the file, you might initialize them all to –1. Then you read through the file, and each time you encounter the start of a new section you determine which field it is, and set the corresponding pointer to the current position of the file pointer in the file.

Once you've passed through the whole file this way, you go to another routine that sets everything to a default value, and then looks at the pointers in a fixed order. You look for the section you want, and if it's there, you jump to it, read it, and overwrite the default. Then you do the same thing for the next section and the next, until you've read everything. Then you can evaluate all of this data to create the weaving specification.



6 Counting up the number of cells required to draw the weaving window.

The other way to go is simpler but slower. As before, you initialize all your data to defaults, and then look for each section of the file in a fixed order, but you simply reread the file from the start each time you need a new section. This is slightly easier to write and debug. I wrote my first WIF reader this way, with the intention of turning it into the more efficient version once I got it working. But WIF files are typically pretty small, and I found that even this inefficient approach read almost every WIF file essentially instantaneously, so I left it that way.

Parsing AWL expressions turned out to be easy, largely because I defined it as a postfix language. I split up the AWL expression into tokens separated by blank spaces, and read each token one at a time, with no lookahead. If the token isn't a keyword (or a symbolic shortcut for one), it goes at the end of the list that's currently on top of the stack. If the token is a keyword, then I pop the necessary arguments off the stack, process them, and push the result back on top. When I'm done parsing a valid expression, there's only one thing on top of the stack: a list of elements that's ready to be copied over into the main weaving window. You can get a copy of my AWL parser written in C# for free from my Web site at http://www.glassner.com. ■

Further Reading

You can find thousands of traditional tartan plaids online. One good place to start is http://www.house-of-tartan.scotland.net, where you can search for tartans based on a sequence of colors, or their traditional associations with Scottish clans, districts, and regiments. You can find another list of tartans, complete with photos of woven examples, at http://www.shetlandpiper.com/tartan_finder. An extensive collection is available at http://www.scottish-tartans-society.org, where you not only can see a picture of each tartan, but the explicit color sequence for weaving it.

You can find the definition for the WIF file format, as well as some samples in that format, online at http://www.mhsoft.com/wif/wif.html.

You can get a free, public-domain copy of the C# source code for my AWL parser in the file AWL.cs from my Web site, http://www.glassner.com.